

FrameForge PRO

Technical overview for external review

A parametric CAD tool for CNC/laser aluminium enclosures,
built by a mechanic (no prior coding background) together with AI assistants.

September 2026

1. What this is

FrameForge PRO is a browser-based parametric CAD tool for designing aluminium enclosures (amplifier chassis and similar boxes) that get manufactured on a CNC mill or laser cutter. It goes from a handful of numbers (internal width/height/depth, fastener size) to a fully dimensioned 6-panel box with corner/support hardware, then exports production files.

The project is a real, ongoing collaboration between a working mechanic (the owner — no prior programming background) and several AI coding assistants over a few months. The owner supplies domain knowledge, real manufacturing constraints and hands-on testing; the AI supplies code and math. You are looking at this because he wants an experienced developer's honest opinion on the result.

What we'd like from you

Read the code with a critical eye — architecture, obvious anti-patterns, anything you would flag in a real code review. Then, if you have a moment, try building a box yourself end-to-end (see section 6) and tell us where the UX confused you. Both kinds of feedback are genuinely useful, don't hold back.

2. Stack and constraints

Layer	Choice
Backend	PHP 8.1, no framework. SQLite (PDO) for project storage.
Frontend	Vanilla JavaScript (no build step, no bundler, no framework). HTML5 canvas for the 2D editor, Three.js (r128) for the 3D preview.
STEP export	Server-side, via headless FreeCAD (freecadcmd) — chosen specifically because it does true solid boolean subtraction (box minus cylinder/cone), not sketch-extrude, which matters for real CNC toleranced parts (see 4.3).
Hosting	Shared Ubuntu 22.04 VPS, nginx + PHP-FPM. No CI/CD, no containers, no package manager for JS (deliberately — see 4.1).
i18n	4 languages (RU/EN/CS/DE), flat key-value dictionaries per language, applied client-side via a small custom <code>`_L()'/`data-i18n`</code> helper.

There is no test suite, no linter config, and no formal build pipeline. This is a conscious trade-off for a single-owner tool developed iteratively through AI pair-programming sessions rather than a deliberate engineering stance — worth keeping in mind when judging code style consistency across files (different files were touched by different AI sessions at different times, sometimes without full-repo visibility).

3. Module map

File	Responsibility
cad_auto.js	Math core — generates all 6 panels + corner/support cube geometry from the box's outer dimensions and wall thicknesses.
ui_hub.js	Single delegated event router for the whole UI (click/mouseover/mouseout on `document`). Replaced three older files (ui_events.js, ui_db.js, element_manager.js) that are now dead.
canvas_mouse.js	Mouse interaction on the 2D canvas — drag/resize/select, undo snapshots.
render_engine.js	2D canvas drawing of the current panel.
viewer_core.js / viewer_utils.js	Three.js 3D assembly preview + per-cube bolt-hole match verification (green/red).
long_panels/lens_*.js, ui_lens.php	The "engineering loupe" — a magnifier overlay for sub-mm precision clicking, physically separate canvas + its own coordinate math.
api.php	Panel save/load/delete (PDO, prepared statements throughout).
generator.php / generator_all.php	Generate Fusion 360 Python scripts (1 panel / all 6).
gen_step.php	Generate a STEP assembly server-side via freecadcmd (see 4.3).
gen_scad.php	DXF export for laser cutting (contour / hatch-fill / surface marking).
gen_cubes.php	Fusion 360 Python script for the corner/support cube hardware.
_hole_finish.php	Shared countersink/counterbore geometry — used by all four exporters above so the numbers can't drift apart between formats.

4. Notable design decisions

4.1 — Absolute vs. display coordinates

Element coordinates are always stored panel-absolute (origin at panel corner), in every code path — database, save/load, every exporter. A recently added "reference point" feature lets the user place an anchor and see $\Delta X/\Delta Y$ (or a live corner/point toggle near the loupe and canvas) computed purely for display. The conversion lives in one function (``window.ffGetDisplayCoords``); nothing downstream of storage/export ever sees relative coordinates. This was a deliberate separation to avoid two sources of truth for geometry.

4.2 — Duplicated computation as a recurring bug source

The single biggest class of bug found during recent hardening passes was the same calculation (bolt count, cube count, panel min-edge-clearance) reimplemented separately in 3+ places (on-screen summary, print specification, pre-export confirmation dialog) that had quietly drifted apart — e.g. one copy deduplicated cubes by a random per-marker id instead of the semantic cube id, overcounting corner cubes 2-3x; another counted ANY hole regardless of diameter as a fastener bolt. All three are now fixed and, where practical, refactored to share one function instead of three copies. If you spot more of this pattern elsewhere, that is the single most valuable thing you could flag.

4.3 — Why STEP export uses solid booleans, not sketch-extrude

Earlier attempts to extrude a flat 2D profile with holes already cut into it (in Fusion 360) produced arcs faceted into dozens of micro-segments on the factory's CNC — enough to cause rejects. ``gen_step.php`` instead builds each panel as ``Part.makeBox()`` and subtracts ``Part.makeCylinder()`` / ``Part.makeCone()`` per hole/countersink — true analytic B-rep primitives from OpenCASCADE (via headless FreeCAD), so arcs are stored as exact circles in the resulting STEP file, not polylines. This is unverified beyond one owner-run smoke test on the actual server (box-minus-cylinder-minus-STEP-export); a real factory run on a produced part is still pending as of this writing.

4.4 — Auth model

Single master password + `hash_equals()` timing-safe comparisons throughout (login, CSRF, the hidden admin-panel entry gate). No user accounts, no roles — appropriate for a single-operator tool, not something to extend without adding real session/role infrastructure first. No hardcoded credentials, API keys or private key material found in the codebase during this review pass — secrets live in a gitignored file/SQLite row, generated on first run.

5. Known open issues

- Undo does not fully cover cube drag. Moving a corner/support cube by mouse synchronises its "twin" holes on 2-3 other panels plus its 3D position. Undo currently only snapshots the panel you're looking at, so undoing a cube move can leave other panels and the 3D view out of sync with what's shown on screen. There is a partial safety net (the pre-export 3D bolt-match check will show red on mismatch), but the underlying gap is not fixed — it needs the undo stack to capture cross-panel state for this one action type, which is a bigger change than a point fix.
- STEP export has not been factory-verified yet (see 4.3) — the geometry approach is sound in principle and passed a local smoke test, but no physical part has been cut from it as of this writing.
- No automated tests anywhere. All verification so far has been manual/live testing by the owner plus code-reading by AI assistants across sessions. If you want to contribute one high-leverage thing, a small test harness around the pure-math files (`cad_auto.js`, `cad_math.js`, `_hole_finish.php`) would probably catch regressions earlier than anything else.

6. Trying it yourself

The fastest path to a finished box, if you want to poke at the UX:

- Open the tool, choose Quick Start, enter three internal dimensions (mm) and pick a fastener size, confirm.
- You land in Auto-Box mode with all 6 panels already generated and corner hardware placed. Switch panels via the "Side" dropdown.
- Try adding a hole/window/vent from the left toolbar, or pick something from the connector cutout library (a dropdown of pre-measured standard connectors — IEC inlet, XLR, 6.35mm jack, RCA, toggle switch, potentiometer).
- Open 3D Preview — green cubes mean every bolt hole lines up correctly across panels; red means it doesn't (drag a cube and re-check to see this live).
- Try Export → STEP without Fusion for a single downloadable assembly file, or the Fusion 360 Python script exports if you have Fusion available.

Full user-facing manual

A separate, much longer PDF walks through every feature for a non-technical user (mechanics, hobbyists) — ask the project owner for `FrameForge_PRO_Guide.pdf` if you want the exhaustive version instead of this developer-oriented summary.

This document was written by Claude (Anthropic) at the project owner's request, specifically for external code review. Happy to answer follow-up questions if you send them back through the owner.